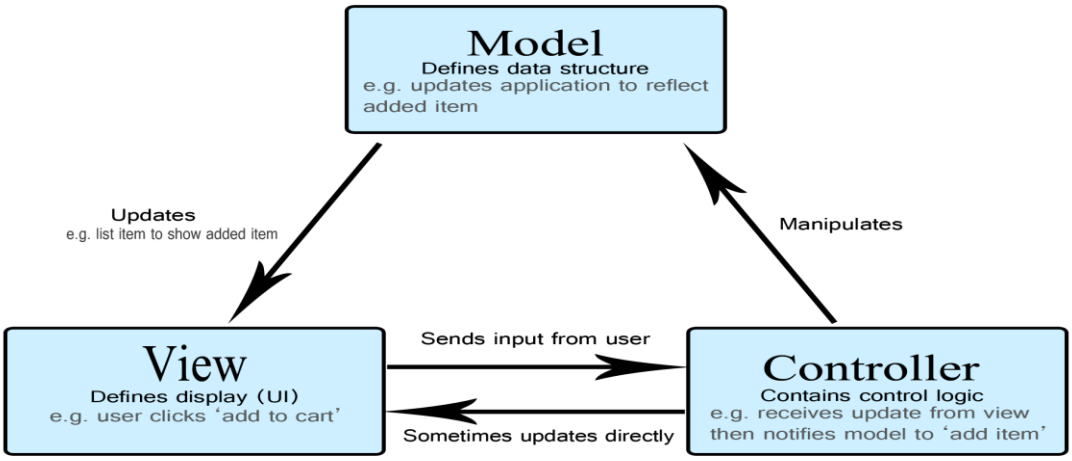
MVC pattern for E-commerce system

MVC: The MVC design pattern is a software architecture pattern that separates an application into three main components: Model, View, and Controller. The Model View Controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information.



Here we Implemented MVC pattern on Node. Js for our project.

Model: The Model component in the MVC (Model-View-Controller) design pattern represents the data and business logic of an application.

In the backend (Node.js), these models are typically associated with database schemas or data structures, serving as representations of fundamental data entities within the context of our e-commerce project. These entities encompass products, users, orders, and other relevant components essential for managing and organizing data effectively. For example:

```
// Product Model
class Product {
  constructor({ id, name, price, description }) {
    this.id = id;
    this.name = name;
```

```
    this.price = price;
    this.description = description;
  }
}

module.exports = Product;
```

Views: Views displays the data from the Model to the user and sends user inputs to the Controller for processing. It is passive and does not directly interact with the Model. For example:

```
import React from 'react';

const ProductView = ({ product }) => {
  return (
    <div>
      <h2>{product.name}</h2>
      <p>${product.price}</p>
      <p>{product.description}</p>
      <button>Add to Cart</button>
    </div>
  );
};

export default ProductView;
```

Controller: Controller acts as an intermediary between the Model and the View. It handles user input and updates the Model accordingly and updates the View to reflect changes in the Model. It contains application logic, such as input validation and data transformation. For example:

```
const products = [
  { id: 1, name: 'Product 1', price: 120.00, description: 'Description of Product
1' },
  { id: 2, name: 'Product 2', price: 200.50, description: 'Description of Product
2' },
  { id: 3, name: 'Product 3', price: 350.00, description: 'Description of Product
3' }
];
```

```
const getAllProducts = (req, res) => {
  res.json(products);
};

const getProductById = (req, res) => {
  const productId = parseInt(req.params.id);
  const product = products.find(p => p.id === productId);

  if (product) {
    res.json(product);
  } else {
    res.status(404).json({ message: 'Product not found' });
  }
};

module.exports = {
  getAllProducts,
  getProductById
};
```

Interactions

In addition to dividing the application into these components, the model–view–controller design defines the interactions between them-

The model is responsible for managing the data of the application. It receives user input from the controller.

The view renders presentation of the model in a particular format.

The controller responds to the user input and performs interactions on the data model objects. The controller receives the input, optionally validates it and then passes the input to the model.